txcasproxy Documentation

Release 0.3

Carl Waldbieser

November 30, 2016

Contents

Contents:

CHAPTER 1

The Twisted CAS Proxy

txcasproxy is a reverse authenticating proxy that is capable of authenticating a user with an instance of the Central Authentication Service (CAS). The proxy then passes a special header (**REMOTE_USER**) to the proxied target, which allows application code to retrieve the user ID of the authenticated user.

txcasproxy is not unlike the combination of the Apache web server using the module mod_auth_cas. The main difference is that *txcasproxy* is a dedicated proxy. This highly specialized nature means that its configuration is rather straightforward. In many cases, the proxy can be configured with just a few command line options.

Because it uses an event-driven reactor rather than a thread based approach, *txcasproxy* can sustain a high volume of connections.

txcasproxy can be customized via the Twisted plugin system. Plugins written in Python can be included when running the proxy from the command line.

CHAPTER 2

Options

```
Usage: twistd [options] casproxy [options]
Options:
      --help-plugins            Help about available plugins.
  -d, --debug                   Errors served as HTML.
  -v, --verbose                 Verbose logging.
      --logout-passthrough      Pass the logout request through to backend
                                service prior to intercepting and redirecting.
  -e, --endpoint=               An endpoint connection string.
  -p, --proxied-url=            The base URL to proxy.
  -c, --cas-login=              The CAS /login URL.
  -s, --cas-service-validate=   The CAS /serviceValidate URL.
  -l, --cas-logout=             The CAS /logout URL.  Requires `logout` option
                                to be set.
  -H, --header=                 The name of the header in which to pass the
                                authenticated user ID. [default: REMOTE_USER]
      --fqdn=                   Explicitly specify the FQDN that should be
                                included in URL callbacks.
  -a, --auth-info-endpoint=     Endpoint for the authentication info service.
  -A, --auth-info-resource=     Resource on the main site that provides
                                authentication info.
      --help-plugin=            Help or a specific plugin.
  -t, --template-dir=           Folder containing templates.
  -T, --template-resource=      Base resource for templates. [default:
                                /_templates]
  -S, --session-length=         Session length in seconds. [default: 900]
  -P, --proxy-client-endpoint=  An endpoint connection string for the proxy web
                                client.
  -C, --cas-client-endpoint=    An endpoint connection string for the back
                                channel CAS web client.
      --help                    Display this help and exit.
      --plugin=                 Include a plugin.
      --version                 Display Twisted version and exit.
      --addCA=                  Add a trusted CA public cert (PEM format).
      --exclude=                Exclude a specific resource from being proxied.
  -L, --logout=                 Add a logout resource pattern to intercept and
                                terminate the proxy session.
      --excludeBranch=          Exclude a resource and all its children from
                                being proxied
```

2.1 Endpoint Specifications

Endpoints are string descriptions of a socket connection a client or server makes. For more details, see the Twisted endpoints documentation.

2.2 TLS Options

Whenever the software attempts to make an HTTPS connection to a proxied site or to a CAS service, it must establish a trust model based on certificates that are signed by some authority. By default, the service uses the platform's underlying certificate authority (CA) trust store. You can extend the default trust store using the `addCA` option.

You can exercise complete and independent control over the trust stores for the CAS web client and/or the proxy web client by specifying client TLS endpoints. Client endpoints allow you to control many aspects of the underlying connection, including client certificates, trust roots, and even the type of underlying connetion (e.g. TCP IP address or UNIX domain socket).

Note: Client endpoints do *not* work in conjunction with the `addCA` option. That option only affects the default web client.

2.2.1 More on Client Endpoints

Because client endpoints must necessarilly specify the connection details, the scheme, host, and port of the URL become superfluous. When using client endpoints it is therefore permissable (and preferable) to omit the URL scheme, host, and port. E.g. '///cas/serviceValidate' rather than 'https://cas.example.net:443/cas/serviceValidate'. In fact, if a client endpoint is used, those parts of the actual URL will be ignored when retieving the resource.

2.3 The REMOTE_USER Header

The reverse proxy attempts to transmit the authenticated user name to the proxied web site as the HTTP header, 'REMOTE_USER'. Some web servers will modify or reject certain headers.

For example, Apache2 will discard 'REMOTE_USER'. It will convert 'Remote-User' into 'HTTP_REMOTE_USER' by the time it reaches the proxied site.

The `--header` option lets you specify the name of the header to pass on to the proxied site.

2.4 Ending the Session

The `logout` option allows you to specify a URL pattern that will be intercepted by the proxy and cause it to terminate its authenticated session. This option may be specified multiple times.

The logout pattern may be a regular URL less the scheme and netlocation. Additionally, the path may include globbing meta-characters.

A query string does not need to be supplied in the pattern. In this case, *any* query string will match the pattern. This is also the case if the entire query string for the pattern is '*'. If the pattern query string is '!', then a URL will *only* match if it has no query string (or an empty query string).

If the pattern contains query string parameters, then a URL will *only* match if it contains *all* the query parameters and values specified in the pattern. A URL *may* contain addtional query string parameters and still match.

If the `cas-logout` URL option is also specified, an HTTP redirect is issued to that URL to terminate the SSO session.

If the `cas-logout` option is not specified, the proxy will reverse proxy the resource (it is not protected). This can be useful if your application allows you to specify a logout URL which you can point to the CAS logout URL. This allows the application to perform its own session termination before the SSO session is ended. It is also useful if the service does not participate in an SSO session but simply uses a CAS service to authenticate.

The `logout-passthrough` option can be used to alter the `cas-logout` behavior. The initial request will be passed through to the proxied service, but its response will be silently discarded. The proxy will issue a response to redirect the requesting agent to the CAS logout URL. This is useful if you require the proxied service to terminate its own local session in addition to terminating the CAS session.

2.5 Authentication Information Service

If you specify an endpoint for the `auth-info-endpoint` option, a web site will be created at that endpoint. The site responds to HTTP GET requests for resources of the form */$USERNAME*, where *$USERNAME* must be a user name that has authenticated with the proxy. The response will be a JSON document that maps attribute names to lists of values.

Note: Even attributes that are single-valued have their values expressed as a list with a single element.

The intention is that access to this site should be limited to the protected service (e.g. with a host based firewall). The protected service can then use this site to retreive attributes for authenticated users using a simple RESTful web service.

2.6 Authentication Information Resource

The `auth-info-resource` can be used to specify a resource on the main site which will respond with a JSON document containing mappings for *username* and *attributes*. The *attributes* key maps to an attribute map identical to the one provided by the authentication information service.

This resource is valid only for requests associated with an already authenticated session. It is therefore more convenient for a client which has authenticated with the proxy to access than for code from the protected service.

2.7 Error Handling

The `debug` option causes any *unexpected* errors (i.e. bugs) to be output to HTML.

There are two expected error scenarios when the proxy may be required to display its own content. If a browser presents a URL to the proxy which contains a CAS service ticket that fails validation, the proxy will emit a 403 (Forbidden) HTTP response code. By default, no content is included.

The second case is when something external to the proxy has gone wrong (perhaps the CAS service is unavailable). In this case, a HTTP 500 response code is returned by the proxy. Again, there is no content by default.

You can provide custom error pages by specifying the `template_dir` option. This should be the path to a folder that contains subfolders `static` and `error`. The `error` folder should contain templates `403.jinja2` and

`500.jinja2`, which should be Jinja2 templates. These templates can access the HTTP request object as the name *request*. The `static` folder may contain any static assets required for rendering the final HTML pages (e.g. images, stylesheets, scripts). These will be served from */_templates/static* by default. You can change the root resource with the `template-resource` option. The name *static_base* is made available to the templates and can be used as a prefix for static resources (the prefix includes a trailing slash).

Note: Only the top-level resource can be changed. For example, if you change the resource to */foo*, the content will be served from */foo/static/*.

CHAPTER 3

Example Integrations

The following are examples for integrating *txcasproxy* with different web technologies.

3.1 Apache + PHP

```php
<?php
# txcasproxy PHP integration example.
?>
<html>
<head>
<title></title>
</head>
<body>
<h2>Server Variables</h2>
<pre>
<?php
print htmlspecialchars(print_r($_SERVER, $return=TRUE));
?>
</pre>
<?php
$username = $_SERVER['HTTP_REMOTE_USER'];
if($username)
{
?>
<h2>txcasproxy CAS Authenticated User Info for <strong><?php echo htmlspecialchars($username);?></str
<pre>
<?php
    $ch = curl_init();
    // set url
    curl_setopt($ch, CURLOPT_URL, "http://127.0.0.1:9444/$username");
    //return the transfer as a string
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    // $output contains the output string
    $output = curl_exec($ch);
    // close curl resource to free up system resources
    curl_close($ch);
    $jsonIterator = new RecursiveIteratorIterator(
        new RecursiveArrayIterator(json_decode($output, TRUE)),
        RecursiveIteratorIterator::SELF_FIRST);

    foreach ($jsonIterator as $key => $val)
```

```
    {
        if(is_array($val))
        {
            echo htmlspecialchars("$key:\n");
        }
        else
        {
            echo htmlspecialchars("$key => $val\n");
        }
    }
?>
</pre>
<?php
}
else
{
?>
<h2>No REMOTE_USER</h2>
<?php
}
?>
</body>
</html>
<?php
?>
```

Full example: `proxy-test.php`

Serve *proxy-test.php* from Apache (or some other web server that will server up PHP). Protect the URL with txcasproxy. Browse to the web page to see info about the logged in user and associated attributes.

Example command line:

```
$ twistd -n casproxy \
    -e 'ssl:9443:certKey=/path/to/server.crt.pem:privateKey=/path/to/server.key.pem' \
    -c 'https://cas.example.net/login' \
    -s 'https://cas.example.net/serviceValidate' \
    -p 'http://protected.example.org/' \
    -a 'tcp:9444' \
    -H 'Remote-User'
```

CHAPTER 4

Plugins

4.1 Access Control

This plugin provides some basic access control based on CAS attributes provided to the proxy during ticket validation. Attribute matching is based on a configuration file in YAML format. The top-level keys of the configuration are attributes that must have been released to the proxy for the authenticated user. If those attributes are not released. If the presence of an attribute is sufficient, then the literal *null* may be used for the value.

If an *allowed_values* key is present as the value, then a list of allowed values is expected to follow. If *any* of the values matches a corresponding value in the released attributes, then that test is passed.

If one or more tests are failed, then access will be denied, and the user-agent will be presented with the 403 ("Forbidden") template as a response.

Example:

```
---
memberOf:
    allowed_values:
        - cn=authorized,ou=groups,dc=example,dc=org
objectClass: null
```

In the above example, the attributes 'memberOf' and 'objectClass' must both have been released to the proxy during ticket validation or the authenticated user will be denied access. Further, the 'memberOf' attribute must have at least one value that matches 'cn=authorized,ou=groups,dc=example,dc=org'.

4.2 Indices and tables

- genindex
- modindex
- search